(12)   **EUROPEAN PATENT APPLICATION**

(72) Inventors:
• Herzberg, Amir
Bronx, New York 10471 (US)
• Krawczyk, Hugo M.
Bronx, New York 10463 (US)

• Kutten, Shay
Rockaway, New Jersey 07866 (US)
• Van Le, An
Sunnyvale, California 94087 (US)
• Matyas, Stephen M.
Poughkeepsie, New York 12601 (US)
• Yung, Marcel M.
New York 10025 (US)

(54)   **Method and system for the secured distribution of programs**

(57)   A method and system for detecting authorized programs within a data processing system. The present invention creates a validation structure for validating a program. The validation structure is embedded in the program and in response to an initiation of the program, a determination is made as to whether the program is an authorized program. The determination is made using the validation structure.
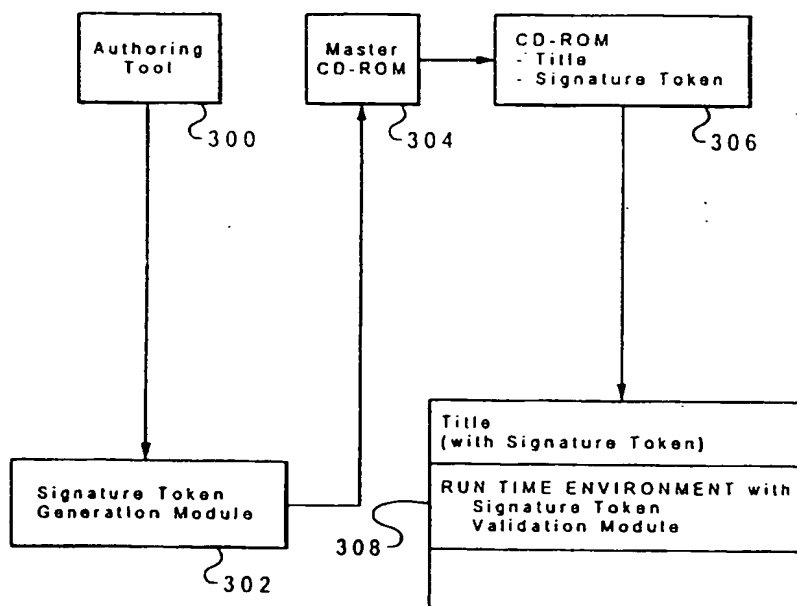
Fig. 3

Description

The present invention generally relates to an improved data processing system, and in particular to a method and system for distributing programs. Still more particularly, the present invention relates to a method and system for checking for authorized programs and detecting unauthorized programs in a data processing system.

Multimedia data processing systems present information in data to a user utilizing sound, graphics, animation, and text. Programs presenting data and information to a user in this form are also called multimedia titles. Typically, a software company develops and markets a software system for the production and presentation of multimedia titles. Such a software system is used in composing multimedia scripts for multimedia titles. Typically, the software system includes a set of authoring tools for producing multimedia titles by developers and a Run Time Environment (RTE) for presenting the multimedia titles to end users. Typically, the RTE is designed to execute on various computing platforms. which makes the authoring tools for the software system desirable to developers. Typically, developers pay a royalty to the software company for using the authoring tools to develop multimedia titles to run on the RTE. But some unscrupulous developers may produce unauthorized titles and avoid royalty payments in producing multimedia titles for use on the RTE.

This invention is directed to providing a method and system for validating programs to be executed on a data processing system by detecting attempted execution of unauthorized programs.

To achieve this, the invention provides a method for enabling a computer to validate programs to be executed therein, the method comprising: creating a validation structure for validating a program, wherein the structure includes data derived from data selected from the program, which is used to determine whether the program is an unauthorized program: imbedding the validation structure in the program: and responsive to an initiation of the program. determining whether the program is an unauthorized program using the validation structure.

This invention provides a method and system for checking for authorized programs and detecting unauthorized programs in a data processing system.

In a preferred embodiment, the present invention provides a method and system for detecting authorized multimedia programs within a data processing system. The present invention creates a validation structure for validating a multimedia program. The validation structure is embedded in the multimedia program and in response to an initiation of the multimedia program, a determination is made as to whether the multimedia program is an authorized multimedia program. The determination is made using the validation structure.

In creating the validation structure, sections of the program (hereinafter called data objects) are selected

and a cryptographic hash value is created or calculated on each of the selected data objects. The cryptographic hash value and the location of the selected data object are stored as a data record within the validation structure. In addition, a signature is included or associated with the validation structure. Preferably, the signature is calculated on the validation structure using a public key cryptographic algorithm.

Determining whether a multimedia program is an authorized multimedia program is accomplished by selecting a subset of the data objects within the multimedia program and validating the selected data objects using the validation structure stored in the multimedia program. This includes the steps of randomly selecting a portion of the data objects from among a defined set of data records listed in the validation structure, reading the selected data objects from the multimedia program using location information stored in the validation structure. and validating the selected data objects using validation information stored the validation structure. For each selected data object, the location information stored in the validation structure is accessed and used to read the selected data object from the multimedia program. A cryptographic hash value is calculated on the selected data object and then compared for equality with a corresponding hash-value-of-reference stored in the validation structure. The hash values must be equal for the selected data objects to be valid. In addition. the validation structure is itself validated through the use of the signature previously calculated on the validation structure, using a public key cryptographic algorithm, and stored within the validation structure. If the signature, validation structure, and subset of selected data objects are valid. the multimedia program is considered to be an authorized multimedia program. An authorized multimedia program is allowed to execute normally. otherwise, execution of the multimedia program may be prohibited or limited execution of the multimedia program may be allowed in response to a determination that the multimedia program is not an authorized program.

The invention will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 depicts a data processing system in the form of a personal computer:

Figure 2 is a schematic diagram of a personal computer system:

Figure 3 is a block diagram of a creation and distribution process for multimedia titles on CD-ROM:

Figure 4 is a depiction of entries in a table of contents:

Figure 5 is a block diagram of a signature token generation module:

Figure 6 is a block diagram of a signature token validation module:

Figure 7 is a flowchart of a process for generating

signature tokens in a signature token generation
module; and
Figure 8 is a flowchart of a process for validating
multimedia titles in a validation program.

With reference now to the figures and in particular
with reference to **Figure 1**, a data processing system,
personal computer system **10** is depicted. As shown,
personal computer system **10** comprises a number of
components, which are interconnected together. More
particularly, a system unit **12** is coupled to and can drive
an optional monitor **14** (such as a conventional video
display). A system unit **12** also can be optionally coupled
to input devices such as a PC keyboard **16** or a mouse
18. Mouse **18** includes right and left buttons (not shown).
The left button is generally employed as the main selec-
tor button and alternatively is referred to as the first
mouse button or mouse button 1. The right button is typ-
ically employed to select auxiliary functions as ex-
plained later. The right mouse button is alternatively re-
ferred to as the second mouse button or mouse button
2. An optional output device, such as a printer **20**, also
can be connected to the system unit **12**. Finally, system
unit **12** may include one or more mass storage devices
such as the diskette drive **22**.

As will be described below, the system unit **12** re-
sponds to input devices, such as PC keyboard **16**, the
mouse **18**, or local area networking interfaces. Addition-
ally, input/output (I/O) devices, such as floppy diskette
drive **22**, display **14**, printer **20**, and local area network
communication system are connected to system unit **12**
in a manner well known. Of course, those skilled in the
art are aware that other conventional components also
can be connected to the system unit **12** for interaction
therewith. Personal computer system **10** includes a sys-
tem processor that is interconnected to a random ac-
cess memory (RAM), a read only memory (ROM), and
a plurality of I/O devices.

In normal use, personal computer system **10** can
be designed to give independent computing power to a
small group of users as a server or a single user and is
inexpensively priced for purchase by individuals or small
businesses. In operation, the system processor func-
tions under an operating system, such as IBM's OS/2
operating system or DOS. (OS/2 is a registered trade-
mark of International Business Machines Corporation).
This type of operating system includes a Basic In-
put/Output System (BIOS) interface between the I/O de-
vices and the operating system. BIOS, which can be
stored in a ROM on a motherboard or planar, includes
diagnostic routines which are contained in a power on
self test section referred to as POST.

A summary of the operation in general of personal
computer system **10** may merit review. Referring to **Fig-
ure 2**, there is shown a block diagram of personal com-
puter system **10** illustrating the various components of
personal computer system 10. **Figure 2** further illus-
trates components of planar **11** and the connection of

planar **11** to I/O slots **46a-46d** and other hardware of
personal computer system **10**. Connected to planar **11**
is the system central processing unit (CPU) **26** com-
prised of a microprocessor which is connected by a high
speed CPU local bus **24** through a bus controlled timing
unit **38** to a memory control unit **50** which is further con-
nected to a volatile random access memory (RAM) **58**.
While any appropriate microprocessor can be used for
CPU **26**, one suitable microprocessor is the Pentium mi-
croprocessor, which is sold by Intel Corporation. ("Pen-
tium" is a trademark of Intel Corporation.)

While the present invention will be described here-
inafter with particular reference to the system block di-
agram of **Figure 2**, it is to be understood at the outset
of the description which follows, it is contemplated that
the present invention may be implemented with other
hardware configurations of the planar board. For exam-
ple, the system processor could be an Intel 80286,
80386, or 80486 microprocessor. These particular mi-
croprocessors can operate in a real addressing mode
or a protected addressing mode. Each mode provides
an addressing scheme for accessing different areas of
the microprocessor's memory.

Returning now to **Figure 2**, CPU local bus **24** (com-
prising data, address and control components) provides
for the connection of CPU **26**, an optional math coproc-
essor **27**, a cache controller **28**, and a cache memory
30. Also coupled on CPU local bus **24** is a buffer **32**.
Buffer **32** is itself connected to a slower speed (com-
pared to the CPU local bus) system bus **34**, also com-
prising address, data and control components. System
bus **34** extends between buffer **32** and a further buffer
36. System bus **34** is further connected to a bus control
and timing unit **38** and a Direct Memory Access (DMA)
unit **40**. DMA unit **40** is comprised of a central arbitration
unit **48** and a DMA controller **41**. Buffer **36** provides an
interface between the system bus **34** and an optional
feature bus such as the Micro Channel bus **44** ("Micro
Channel" is a registered trademark of International Busi-
ness Machines Corporation.) Connected to bus **44** are
a plurality of I/O slots **46a-46d** for receiving Micro Chan-
nel adapter cards which may be further connected to an
I/O device or memory. In the depicted example. I/O slot
46a has a hard disk drive connected to it; I/O slot **46b**
has a CD-ROM drive connected to it; and I/O slot **46c**
has a ROM on an adapter card connected to it. Other
devices, such as a modem may be connected to an I/O
slot. An arbitration control bus **42** couples the DMA con-
troller **41** and central arbitration unit **48** to I/O slots **46**
and diskette adapter **82**. Also connected to system bus
34 is a memory control unit **50** which is comprised of a
memory controller **52**, an address multiplexer **54**, and a
data buffer **56**. Memory control unit **50** is further con-
nected to a random access memory as represented by
RAM module **58**. Memory controller **52** includes the log-
ic for mapping addresses to and from CPU **26** to partic-
ular areas of RAM **58**. While the personal computer sys-
tem **10** is shown with a basic 1 megabyte RAM module,

it is understood that additional memory can be interconnected as represented in **Figure 2** by the optional memory modules **60** through **64**.

A further buffer **66** is coupled between system bus **34** and a planar I/O bus **68**. Planar I/O bus **68** includes address, data, and control components respectively. Coupled along planar bus **68** are a variety of I/O adapters and other peripheral components such as display adapter **70** (which is used to drive an optional display **14**), a clock **72**, nonvolatile RAM **74** (hereinafter referred to as "NVRAM"), a RS232 adapter **76**, a parallel adapter **78**, a plurality of timers **80**, a diskette adapter **82**, a PC keyboard/mouse controller **84**, and a read only memory (ROM) **86**. The ROM **86** includes BIOS which provides the user transparent communications between many I/O devices.

Clock **72** is used for time of day calculations. NVRAM **74** is used to store system configuration data. That is, the NVRAM will contain values which describe the present configuration of the system. For example, NVRAM **74** contains information which describe the capacity of a fixed disk or diskette, the type of display, the amount of memory, etc. Of particular importance, NVRAM **74** will contain data which is used to describe the system console configuration: i.e., whether a PC keyboard is connected to the keyboard/mouse controller **84**, a display controller is available or the ASCII terminal is connected to RS232 adapter **76**. Furthermore, these data are stored in NVRAM **74** whenever a special configuration program is executed. The purpose of the configuration program is to store values characterizing the configuration of this system to NVRAM **76** which are saved when power is removed from the system.

Connected to keyboard/mouse controller **84** are ports A and B. These ports are used to connect a PC keyboard (as opposed to an ASCII terminal) and mouse to the PC system. Coupled to RS232 adapter unit **76** is an RS232 connector. An optional ASCII terminal can be coupled to the system through this connector.

Specifically, personal computer system **10** may be implemented utilizing any suitable computer such as the IBM PS/2 computer or an IBM RISC SYSTEM/6000 computer, both products of International Business Machines Corporation, located in Armonk, New York. ("RISC SYSTEM/6000" is a trademark of International Business Machines Corporation and "PS/2" is a registered trademark of International Business Machines Corporation.)

Distribution of multimedia programs or titles (hereinafter called "multimedia titles") involves an application developer who produces multimedia titles using an authoring tool and a Run Time Environment (RTE) provided by a multimedia company and a user who purchases multimedia titles for execution on a computer or computer platform executing the RTE. In a preferred embodiment of the present invention, checking for authorized multimedia titles and detecting unauthorized multimedia titles involves a scheme of digital signatures using a

public key algorithm. A "public key" is a key made available to anyone who wants to encrypt information. In public key cryptography, public key algorithms are used in which a public key is used for encryption and a private key is used for decryption. The basis for public key cryptography includes discrete logarithms, factoring, and the knapsack problem. Each authorized multimedia title includes an embedded digital signature token that can be verified by the RTE before the multimedia title is permitted to execute on the data processing system.

Two cryptographic subsystems are employed to facilitate the signature token generation and signature token verification processes. One cryptographic subsystem enables the generation of signature tokens that, when embedded in authorized multimedia titles, will permit these titles to be validated. Another cryptographic subsystem is employed to validate the signature tokens. In this manner authorized multimedia titles may be distinguished from unauthorized multimedia titles.

With reference to **Figure 3**, a block diagram of a creation and distribution process for multimedia titles on CD-ROM is depicted. Those skilled in the art will recognize that the invention could be implemented in a system wherein multimedia titles are distributed on media other than a CD ROM medium. A multimedia title is developed by a developer using authoring tool **300**. The multimedia title is then processed using signature token generation module **302**. This module generates a signature token for the multimedia title. The signature token is embedded within the multimedia title. Thereafter, the multimedia title with the signature token embedded within it is sent back to the developer who creates a master CD-ROM **304**. Alternatively, the signature token and multimedia title are sent back to the developer, whereupon the signature token is embedded into the multimedia title and a master CD-ROM **304** is created by the developer. From master CD-ROM **304**, CD-ROM **306** is produced containing the multimedia title and the embedded signature token. CD-ROM **306** may be placed within data processing system **308**, which includes the RTE with the signature token validation module. When the title is to be executed within data processing system **308**, the RTE reads the signature token from the CD-ROM and validates the signature token and a selected portion of the data objects also read from the CD-ROM using the signature token validation module.

Typically, a multimedia title takes about one hour to play and contains about 650 megabytes of data. As a result, it is inefficient to validate a multimedia title by reading and checking each byte within the title. Consequently, the multimedia title is validated by checking a portion of the data contained therein.

In this embodiment, random sampling of data to validate multimedia titles is employed. If the data locations to be sampled were constant from one instance of validation to the next, then only a small portion of the multimedia title would be checked. In such a situation, forged titles could be more easily constructed. But by

randomly selecting data locations for sampling, the possibility of forged multimedia titles is greatly reduced.

In addition, data context sampling is utilized. A significant improvement in the validation of multimedia titles may be achieved if the logical structure of the multimedia titles themselves is employed to identify key pieces of data to be validated. For example, a preferred checking strategy may be based on checking part or all of the data in the table of contents for each file in a multimedia title. A multimedia title consists of one or more files, each containing its own table of contents. In many cases the multimedia title contains only one such file. When a file is opened, the table of contents is the first item to be read.

With reference now to **Figure 4**, a depiction of entries in a table of contents is illustrated. Table of contents **400** includes entries **402-408**. Each entry includes an object identifier, a property identifier, a type, and a location (offset and length). As a result, a particular entry indicates that at a particular offset on the disk. for this many bytes, a property with this ID belonging to an object with this ID of this type is located. Because the table of contents references data on the basis of an Object ID and a Property ID, the referenced data object is said to be referenced by an object-property (OP) pair and the data object is referred to as OP data. Of course, other formats and specifications for the table of contents may be utilized. The table of contents. regardless of its format, structure, and sematics, may be employed to effectively validate a multimedia title. Typically in multimedia titles, the table of contents is an example of a relatively short and easily identifiable piece of information that has an intrinsic dependency with most of the other parts of the multimedia title. The table of contents could take the form of a symbol table, a linkage map, and so forth, but is rigidly specified and highly structured.

Furthermore, for multimedia titles, the first few displayed screens typically contain the name of the title and its version. As a result, protecting these screens is desirable. Therefore, a checking strategy may include checking the first few screens of data displayed to a user so that a forged title, whose name is for example "Demons for the Deep", would be forced to display the name of the title upon which it is piggy-backing, say "Desert Wargames".

The present implementation provides a method and system for validating multimedia titles by validating part or all of the table of contents and the first few displayed screens containing the name of the title and its version for each multimedia file and validating a subset of the data objects in the multimedia title. These data objects are selected randomly. But those skilled in the art will recognize that the data objects could be selected using a method which is non-random.

With reference to **Figure 5**, a diagram of a signature token is depicted. Signature token **500** is constructed by a signature token generation module (not shown in **Figure 5**). The signature token is constructed step-by-step

by making repeated service requests to the signature token generation module. Once created, signature token **500** is embedded in the multimedia title upon which it was generated. This signature token is validated by a signature token validation module in the RTE. In validation, the signature token is validated step-by-step by making repeated service requests to the signature token validation module.

Signature token **500** includes a header **502** and data records 1 through n that correspond to data or data objects in the multimedia title that can be selected and validated. The data records 1 through n in the signature token are different from the data objects in the multimedia title, although there is a direct correspondence. In addition, signature token **500** includes digital signature **504**, which is employed to validate the header and the series of data records 1 through n in the signature token. Each data record within signature token **500** includes location specific information, L, and a cryptographic hash value, H. Location specific information tells the signature token validation module the location or locations in the multimedia title of the data to be validated. The hash value is calculated on the specific multimedia data referenced by L. In this embodiment, the cryptographic hash value is calculated using a one-way function. A one-way function is one where it is computationally infeasible to find two different inputs X and Y, such that the cryptographic hash of X is equal to the cryptographic hash of Y. The term "computationally infeasible" is one used in prior art to describe a mathematical procedure that cannot be performed in the practical sense because of the very large number of computational steps required. However, the term is not precise in that there is no prescribed number of computations above which a computation is said to be computationally infeasible and below which the computation is said to be computationally feasible. In general, a mathematical procedure is said to be computationally infeasible if the cost or time to perform the necessary computations is beyond reasonable human means, e.g., if all the computers in the world linked together couldn't solve the problem in a billion billion billion years.

Location specific information may be a combination of location specific information stored in signature token **500** and location specific information derived algorithmically. The table of contents in a multimedia title is one example of location specific information that can be omitted from signature token **500**. In other words, signature token **500** does not need to store an address or location of the table of contents because a simple procedure exists for always finding the table of contents given a standard starting point.

A portion of the data records 1 through n in signature token **500** will reference data in one or more data objects in the multimedia title. The signature token generation module employs a process to select a subset of different data objects (referenced by the object-property pairs in the table of contents) from a multimedia title to

be validated. This process will be described in more detail below. It is desirable to have an element of randomness in this process although strictly speaking random in this process is not required. Once the subset of data objects in the multimedia title have been identified, a subportion of the data in each data object is selected and the locations of these subportions of data (denoted L1, L2, ..., Ln) are used by the signature token generation module to read the data associated with each subportion. The signature token generation module then issues a service request to the cryptographic subsystem to generate a cryptographic hash value Hi on each subportion of data referenced by location information Li. The cryptographic hash value, Hi, is then stored in data record i together with the location information, Li. After the header and data records 1 through n have been created the signature token generation module issues a service request to the cryptographic subsystem to calculate a cryptographic hash value on the signature token. except for the digital signature. The signature token generation module then issues a service request to the cryptographic subsystem to calculate a digital signature on the signature token, using the cryptographic hash value calculated on the signature token and the private key of the multimedia company. The digital signature is then stored in the signature token.

The signature token validation module will randomly select and validate a subset of the data records in the signature token thus introducing an element of randomness into the validation process. Once the subset of data records has been randomly selected, the signature token validation module will process each data record on a record-by-record basis. The location specific information, L, in the data record is used to read the referenced data from the CD-ROM. A service request is then issued to the cryptographic subsystem to generate a cryptographic hash value, H, on the referenced data. This calculated cryptographic hash value is then compared for equality with the reference cryptographic hash value, H, stored in the data record. If the cryptographic hash values are equal, the process then continues. Otherwise, processing may be halted with an indication that validation has failed. After each data record is processed, the signature token validation module issues a service request to the cryptographic system to cryptographically hash the signature token (except for the digital signature) that it previously read from the multimedia title. Then a service request is issued to the cryptographic subsystem to validate the digital signature. The digital signature is encrypted with the public key of the multimedia company, stored in the signature token validation module. The encrypted value of the digital signature contains a hash-value-of-reference previously calculated on the valid signature token. The calculated hash value is then compared for equality with the so-obtained hash-value-of-reference. If the hash values are equal, then the signature token and digital signature are valid. Otherwise, the signature token and digital signature are

not valid. Those skilled in the art will recognize that any of the known digital signature methods may be used.

The digital signature also can be calculated on a cryptographic hash value representing the root of a tree of cryptographic hash values, e.g., a binary tree of cryptographic hash values as described in United States Patent No. 4,309,569, "Method of Providing Digital Signatures". By storing n-1 additional intermediate cryptographic hash values in the signature token, m of the possible n data objects in the signature token can be validated using $m \cdot \log 2n$ hashing operations instead of n hashing operations, which for small n may be more advantageous. A method of calculating the tree of cryptographic hash values is described in United States Patent No. 4,309,569 and in United States Patent No. 5,231,666, "Cryptographic Method For Updating Financial Records".

With reference now to **Figure 6**, a block diagram of a signature token generation module and a signature token validation module is depicted. Signature token generation module **600** includes generation module **602** and cryptographic subsystem **604**. Signature validation module **606** includes a validation program **608** and a cryptographic subsystem **610**. Generation program **602** in signature token generation module **600** selects data (including data in randomly selected data objects) in the multimedia title to be validated. Generation program **602** reads data from the multimedia title and processes the data by issuing repeated service requests to cryptographic subsystem **604**. These repeated service requests to process data are used to build a signature token. Similarly, validation program **608** in signature token validation module **606** randomly selects (for subsequent processing and validation) a subset of the data records in the signature token generated by signature token generation module **600**. This data is read from the CD-ROM and is processed by issuing repeated service requests to cryptographic subsystem **610** to validate the signature token.

Cryptographic subsystem **604** provides the following cryptographic services to generation program **602**: (1) initialize random number generator, (2) generate random number, (3) generate hash value, (4) generate digital signature, and (5) verify digital signature. A verify digital signature function is provided so that once a signature token is generated, generation program **602** can validate the signature token to insure that the signature token can be correctly processed by cryptographic subsystem **610** and signature token module **606**. Such a verification function provides a high integrity process in creating multimedia titles with embedded signature tokens. Cryptographic subsystem **610** in signature token validation module **606** supports validation program **608** by providing the following services: (1) initialize random number generator. (2) generate random number. (3) generate hash value. and (4) verify digital signature. The random number generation is employed to randomly select data records in the signature token to be validated

Algorithms and procedures for generating random numbers, generating hash values, and for generating and verifying digital signatures are well known within the prior art.

The random number generator function can be implemented using the pseudo random integer generator supplied in Appendix C of American Standard Institute (ANSI) Standard X9.17, "Financial Institution Key Management (Wholesale)" as specified in Appendix 3 of Federal Information Processing Standard (FIPS) 186.

The initialize random number generator function is a function that causes a secret seed value to be generated within the random number generator or alternatively to be provided as an input to the random number generator. A simple method for initializing the random number generator is to employ a free-wheeling counter in combination with a series of requests to a human who interacts with the initialize random number generator function via a workstation keyboard and display. The user is repeatedly prompted to enter a character at the keyboard. Each time the random number generator function gets control, it reads the free-wheeling counter and this value is combined with a value stored in an accumulator. The final accumulator value is taken as the secret seed value. Because of small differences in human response time, the value of the free-wheeling counter is unpredictable, and therefore the resulting seed value will be random.

The generate hash function can be implemented using one of several algorithms, including the MDC-2 or MDC-4 algorithms described in U.S. Patent No. 4,908,861 entitled "Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function" or using the hashing algorithm in FIPS 180, "Secure Hash Standard."

The generate digital signature and verify digital signature functions can be implemented with the RSA public key algorithm using ISO Standard 9796 "Digital Signature Scheme Giving Message Recovery" or by using signature generation and signature verification algorithms in FIPS 186, "Digital Signature Standard."

Although the depicted embodiment employs a public key algorithm, which is an asymmetric algorithm, the digital signature can be based on a symmetric algorithm such as the Data Encryption Standard (DES).

With reference now to **Figure 7**, a flowchart of a process for generating signature tokens in a signature token generation module is depicted. The process begins by receiving a multimedia title with the signature token (step **700**). Thereafter, the process fixes variables N and M such that N is the maximum number of data records to be created for the signature token that is to be produced, and M is the maximum record data length in bytes of each selected data object (step **702**). For example, N = 1,000 and M = 1,000 are possible values for N and M. The process then locates and reads the table of contents or equivalent information contained in the multimedia title (step **704**). This information is called

"D1". A 128 bit MDC, designated MDC 1, is calculated on the data D1 (step **705**). Using the table of contents as necessary, the first logo screen is located and read from the multimedia title (step **706**). The logo screen contains the name of the multimedia title and is designated "D2". A 128 bit MDC, designated MDC 2, is calculated on the data D2 (step **707**).

Using the table of contents, the process then determines the number of multimedia object-property pairs, S, contained in the multimedia title (step **708**). An object-property is data pointed to by the object-property entry in a table of contents. S is calculated as S = S<1>+S<2>+...+S<T>, where S<i> denotes the number of properties associated with multimedia object "i", and "T" denotes the number of multimedia objects within the multimedia title. The process then sets n = min (N,S), the minimum of N and S (step **710**). Typically, S will be much larger than N so that N = n will generally hold true. The process then sets R = n-2 and X = 1 (step **712**). Thereafter, an object-property pair is randomly selected from the multimedia title: the data corresponding to each object-property pair is located and read, and the locations(s) of the data are saved (step **714**). "D3" denotes the information corresponding to the object-property pair (or the first n bytes of information) in the randomly selected object-property, which ever is less. L3 denotes the information in the table of contents employed to locate and read the data corresponding to the randomly selected object-property pair. In effect, the process selects one of the S object-property pairs (excluding the table of contents and the logo screen) and then at the next iteration randomly selects one of the S-1 remaining different object-property pairs with the previously selected object-property pair being excluded and so on and so forth.

The process then calculates the MDC on the data or data object referenced by the selected object-property pair (step **716**). A 128 bit MDC is calculated on the data associated with the object-property pair using the method described in United States Patent No. 4,908,861 entitled "Data of Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function". The MDC values calculated on values D3 through Dn are denoted MDC 3 through MDCn, respectively, as the process loops through (steps **714** and **716**). The process determines whether X = R (step **718**). If X is not equal to R, the process sets X = X + 1 (step **720**) and returns to step **714**. This continues until R = n - 2 different object property pairs have been processed.

When X=R, the process then proceeds to build a signature token, such as signature token **500** depicted in **Figure 5**, consisting of a header, n data records, and a digital signature (step **722**). The token type (e.g., "multimedia signature token") and n (number of data records) is stored within the header. In the first data record, the data location and the hash value are set to the constant value "table of contents" and the calculated

value of MDC 1, respectively. The term "table of contents" is stored in the data location sub-field in lieu of location information because, in this implementation, the signature token validation module always knows the location of the table of contents in accordance with a preferred embodiment of the present invention. In the second data record in the signature token, the data location and hash value are set to the constant value "logo screen" and the calculated value of MDC2, respectively. The term "logo screen" is stored in the data location sub-field in lieu of location information because in this implementation the signature token validation module is able to calculate the location of the logo screen from the information found in the table of contents. The data locations and hash values in data records 3, 4, ..., n are set to (L3, MDC3), (L4, MDC4) ..., (Ln, MDCn), respectively. Note also that the hash values H1 through Hn in data records 1 through n of the signature token are equal to the MDC values: MDC1 through MDCn, respectively.

Furthermore, the signature token also includes space for a digital signature. An RSA digital signature based on ISO Standard 9796 might typically require from 512 to 2048 bits of storage space. A DSS digital signature based on FIPS 186 requires 320 bits of storage space. At step **722**, the digital signature portion of the signature token is presently uninitialized. Thereafter, the process calculates a digital signature for the signature token with the multimedia company's private key (step **724**). More specifically, a 128 bit MDC is calculated on the portion of the signature token consisting of the header and the n data records. This calculation may be performed using the process described in United States Patent No. 4,908,861 entitled "Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function".

Thereafter, a digital signature is calculated on the MDC value using the private key of the multimedia company. The digital signature could be calculated with an RSA private key in accordance with ISO standard 9796 or with a DSA private key in accordance with FIPS 186. Those skilled in the art will recognize that private keys of different lengths can be employed to calculate the digital signature. and that digital signatures of different lengths can be stored in the signature token. Thereafter, the digital signature is stored in the signature token and the process terminates.

With reference now to **Figure 8**, a flowchart of a process for validating multimedia titles in a validation program is depicted. This process is employed within the signature token validation module **606** of **Figure 6**. The process begins by receiving a multimedia title with the embedded signature token (step **800**). The process then fixes variable R, where R in this flowchart is the number of data records to be randomly authenticated (step **802**). For example. R may be any value. such as 3, 4, 5, etc., but will generally be a small value in order to minimize the required processing time. Thereafter, the process reads the embedded signature token (step

804). The table of contents may be used to locate and read the embedded signature token.

The process then locates and reads the table of contents or equivalent information contained in the multimedia title (step **806**). This information is called "D1". The process then calculates a 128 bit MDC on the table of contents, D1 (step **808**). This calculation may be performed in various ways. For example, the process described in United States Patent No. 4,908,861, entitled "Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function" may be used. The process then validates the MDC calculated on the table of contents against the MDC-of-reference stored in the signature token (step **810**). The 128 bit MDC calculated in step **808** is compared with H1 in the first data record in the signature token. The process then determines whether the two values are equal (step **812**). If MDC is not equal to H1, the process then indicates that the title is invalid (step **814**). An invalid multimedia title so-detected by the signature token validation module causes an indicator or indication to be set, but does not halt processing. However. the processing could equally be halted as soon as an invalid condition is detected.

If, however, MDC is equal to H1. the process then locates and reads the first logo screen that is displayed to a user in the multimedia title (step 816). The table of contents may be used to locate and read this logo screen. The logo screen contains the name of the multimedia title, and this information is called "D2". Next, an MDC is calculated on the data comprising the logo screen (step **818**). A 128 bit MDC is calculated The process then validates the MDC calculated on the logo screen against the MDC stored in the signature token (step **820**). The 128 bit MDC calculated in step **818** is compared with H2 in the second data record in the signature token. The process then determines whether the two values are equal (step **822**). If MDC is not equal to H2, the process then indicates that the title is invalid (step **824**). The process then sets Y = 1 (step **826**).

The process also sets Y = 1 if MDC is equal H2 (step **826**). The process then randomly selects and reads one of the n - 2 remaining data records in the signature token (step **828**). The data records are selected from data record 3 through data record n. inclusively. Each data record is associated with and refers to all or a portion of the data for a single object-property pair in the multimedia title. The process then reads the data OP1 associated with the object-property pair pointed to by the selected data record (step **830**). The data location information in the data record is used to locate and read the data OP1 associated with the referenced object-property pair in the multimedia title.

Next, the process calculates a 128 bit MDC on the selected data OP1 (step **832**). The process then validates the MDC calculated on the data OP1 against the corresponding MDC. or hash value H, stored in the data record read from the signature token (step **834**). In step

834 the 128 bit MDC calculated in step 832 is compared with the corresponding MDC, or hash value H, recovered from the data record. If the calculated MDC and stored MDC are equal, control is passed to step 838. If the MDC values are not equal, the process indicates that the title is invalid (step 836) and control is passed to (step 838). The process determines whether Y = R (step 838). If Y is not equal to R, the process sets Y= Y + 1 (step 839) and returns to step 828. This continues until R data records have been randomly selected from among data records 3 through n in the signature token, and processed. When Y = R, the process then proceeds to validate the digital signature and signature token. Next, the process reads the digital signature in the signature token (step 840).

Thereafter, the process accesses the multimedia company's public key, which is stored as a fixed constant in the signature token validation module (step 842). The process then validates the signature using the public key (step 844). A 128 bit MDC is calculated on the portion of the signature token consisting of the header and the n data records. When an RSA digital signature is employed, the original 128 bit hash value H is recovered from the RSA digital signature using the process described in ISO standard 9796 utilizing the RSA public key. Thereafter, the 128 bit MDC calculated on the signature token including the header and the n data records are compared with the 128 bit hash value H. If these MDCs are equal, then the multimedia title is accepted as valid. Otherwise, the process indicates that the title is invalid. When a digital signature based on FIPS 186 is employed, the signature is validated by following the steps outlined in FIPS 186, which are different from those used to validate an RSA digital signature.

The process then accepts the title or handles the title invalid condition (step 846). If the title is indicated as being invalid, several actions may be taken. For example, the multimedia title may be rejected, the multimedia title may be accepted, but the user is given a warning screen; the multimedia may be accepted, but the user may be required to start over. Of course, the signature may be validated before the table of contents, logo screen, and randomly selected data records are checked. Depending on the implementation other logical structures other than the table of contents or logo screen may or may not be checked.

The processes depicted in **Figures 3-8** may be implemented by those of ordinary skill in the art within the data processing system depicted in **Figures 1** and **2**. The processes of the present invention also may be implemented in a program storage device that is readable by a data processing system, wherein the program storage device encodes data processing system executable instructions coding for the processes of the present invention. The program storage device may take various forms including, for example, but not limited to a hard disk drive, a floppy disk, an optical disk, a ROM, and an EPROM, which are known to those skilled in the art. The

processes stored on a program storage device are dormant until activated by using the program storage device with the data processing system. For example, a hard drive containing data processing system executable instructions may be connected to a data processing system; a floppy disk containing data processing system executable instructions may be inserted into a floppy disk drive in the data processing system; or a ROM containing data processing system executable instructions may be connected to the data processing system via a card or adapter connected to an I/O slot.

While the invention has been particularly shown and described with reference to a preferred embodiment, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the scope of the invention.

There has been described a storage device readable by a data processing system and encoding data processing system executable instructions comprising: selection means for selecting a plurality of sections from within a program: creation means for creating a cryptographic hash value for each selected section from the plurality of selected sections within the program: and storage means for storing the cryptographic hash value and a location value for each selected section as a data record within a validation structure, wherein the location is a location of the selected section within the program, wherein the means are activated when the storage device is connected to and accessed by a data processing system, and a storage device readable by a data processing system and encoding data processing system executable instructions for validating a program, wherein the program includes a validation structure having a plurality of data records, wherein each data record within the plurality of data records includes a cryptographic hash value for a section of the program and a location value, wherein the location value indicates a location of the section, the storage device comprising: creation means for creating a cryptographic hash value on the section in location indicated by the location value for the randomly selected data record for each randomly data selected record: and comparison means for comparing the created cryptographic hash value with the hash value within the randomly selected data record, wherein the means are activated when the storage device is connected to and accessed by a data processing system.

Claims

1. A method for enabling a computer to validate programs to be executed therein, the method comprising:

creating a validation structure for validating a program, wherein the structure includes data derived from data selected from the program,

which is used to determine whether the program is an unauthorized program:

imbedding the validation structure in the program; and responsive to an initiation of the program, determining whether the program is an unauthorized program using the validation structure.

2. A method as claimed in claim 1, wherein the creating step comprises randomly selecting data from within the program.

3. A method as claimed in claim 1 or claim 2, wherein the determining step comprises randomly selecting portions of the validation structure; and determining whether the program is an unauthorized program using the randomly selected portions of the validation structure.

4. A method as claimed in any preceding claim, wherein the creating step comprises:

selecting a plurality of sections from within the program;
creating a cryptographic hash value for each selected section from the plurality of selected sections within the program: and
storing the cryptographic hash value and a location value for each selected section as a data record within a validation structure. wherein the location is a location of the selected section within the program.

5. A method as claimed in claim 4. wherein the creating step further comprises:

creating a signature for the validation structure, wherein the signature is a cryptographic hash value calculated on the validation structure: and
associating the signature with the validation structure.

6. A method as claimed in claim 5, wherein the associating step comprises placing the signature within the validation structure.

7. A method as claimed in any preceding claim, wherein the determining step comprises:

randomly selecting a number of data records from within the validation structure:
for each randomly data selected record. creating a cryptographic hash value on the section in located indicated by the location value for the randomly selected data record; and
comparing the created cryptographic hash value with the hash value within the randomly se-

lected data record.

8. A method as claimed in claim 7. wherein the determining step further comprises:

creating a cryptographic hash value for the validation structure: and
comparing the created cryptographic hash value with the signature.

9. A method in a data processing system for creating a validation structure for use in validating a program, the method comprising:

selecting a plurality of sections from within the program:
creating a cryptographic hash value for each selected section from the plurality of selected sections within the program: and
storing the cryptographic hash value and a location value for each selected section as a data record within a validation structure, wherein the location is a location of the selected section within the program.

10. A method as claimed in claim 9, wherein the selecting step comprises randomly selecting a plurality of sections from within the program.

11. A data processing system for creating a validation structure for use in validating a program. the data processing system comprising:

selection means for randomly selecting a plurality of sections from within the program:
creation means for creating a cryptographic hash value for each selected section from the plurality of selected sections within the program: and
storage means for storing the cryptographic hash value and a location value for each selected section as a data record within a validation structure, wherein the location is a location of the selected section within the program.

12. A method in a data processing system for validating a program, wherein the program includes a validation structure having a plurality of data records, wherein each data record within the plurality of data records includes a cryptographic hash value for a section of the program and a location value, wherein the location value indicates a location of the section, the method comprising:

randomly selecting a number of data records from within the validation structure:
creating a cryptographic hash value on the section in location indicated by the location value

for the randomly selected data record for each
randomly data selected record; and
comparing the created cryptographic hash val-
ue with the hash value within the randomly se-
lected data record. 5

13. A data processing system for validating a program,
wherein the program includes a validation structure
having a plurality of data records, wherein each da-
ta record within the plurality of data records includes 10
a cryptographic hash value for a section of the pro-
gram and a location value, wherein the location val-
ue indicates a location of the section, the data
processing system comprising:

  15

random selection means for randomly selecting
a number of data records from within the vali-
dation structure;
creation means for creating a cryptographic
hash value on the section in location indicated 20
by the location value for the randomly selected
data record for each randomly data selected
record; and
comparison means for comparing the created
cryptographic hash value with the hash value 25
within the randomly selected data record.

30

35

40

45

50

55

Fig. 1

Fig. 2

Fig. 3

Fig. 4

Signature Token:

| Header | Data Record 1 | Data Record 2 | ••• | Data Record n | Digital Signature |

502

500

504

Data Record 1:

| Data Location(s) 1 | Hash Value 1 |

Fig. 5

Signature Token
Generation Module

600

| Generation Program | 602 |

| Cryptographic Subsystem | 604 |

Signature Token
Validation Module

606

| Validation Program | 608 |

| Cryptographic Subsystem | 610 |

Fig. 6

```
                    ┌──────────┐
                    │  Start   │
                    └────┬─────┘
                         │        ⌐700
              ┌──────────▼──────────┐
              │      Receive        │
              │     multimedia      │
              │   title (without    │
              │  signature token)   │
              └──────────┬──────────┘
                         │        ⌐702
              ┌──────────▼──────────┐
              │    Fix variables    │
              │      N and M        │
              └──────────┬──────────┘
                         │        ⌐704
              ┌──────────▼──────────┐
              │   Read table of     │
              │    contents in      │
              │  multimedia title   │
              └──────────┬──────────┘
                         │        ⌐705
              ┌──────────▼──────────┐
              │   Calculate an      │
              │  MDC on table of    │
              │   contents data     │
              └──────────┬──────────┘
                         │        ⌐706
              ┌──────────▼──────────┐
              │  Read logo screen   │
              │   in multimedia     │
              │       title         │
              └──────────┬──────────┘
                         │        ⌐707
              ┌──────────▼──────────┐
              │   Calculate an      │
              │   MDC on logo       │
              │   screen data       │
              └──────────┬──────────┘
                         │        ⌐708
              ┌──────────▼──────────┐
              │ Calculate number    │
              │ of object-property  │
              │    pair(s) in       │
              │  multimedia title   │
              └──────────┬──────────┘
                         │        ⌐710
              ┌──────────▼──────────┐
              │    Set n = min      │
              │      (N, S)         │
              └──────────┬──────────┘
```

$$\text{Set } R = n - 2 \text{ and } X = 1 \quad (712)$$

Select an object-property and read data from the multimedia title. Remember location of referenced data. (714)

Set X = X + 1 (720)

Calculate MDC on data (716)

X = R ? (718) No

Yes (722)

Build a signature token (less the digital signature) (722)

Calculate a digital signature on the signature token and store in signature token (724)
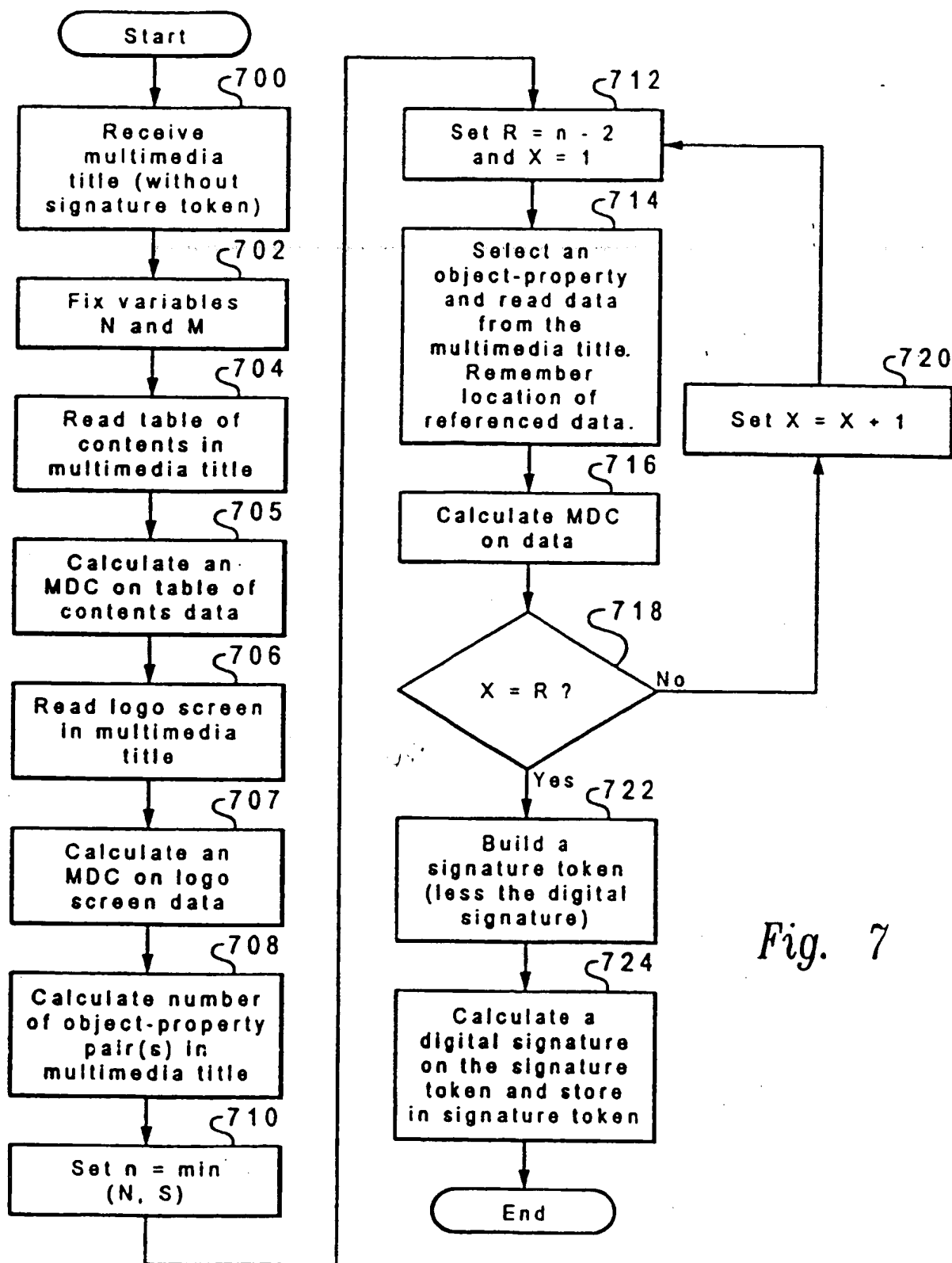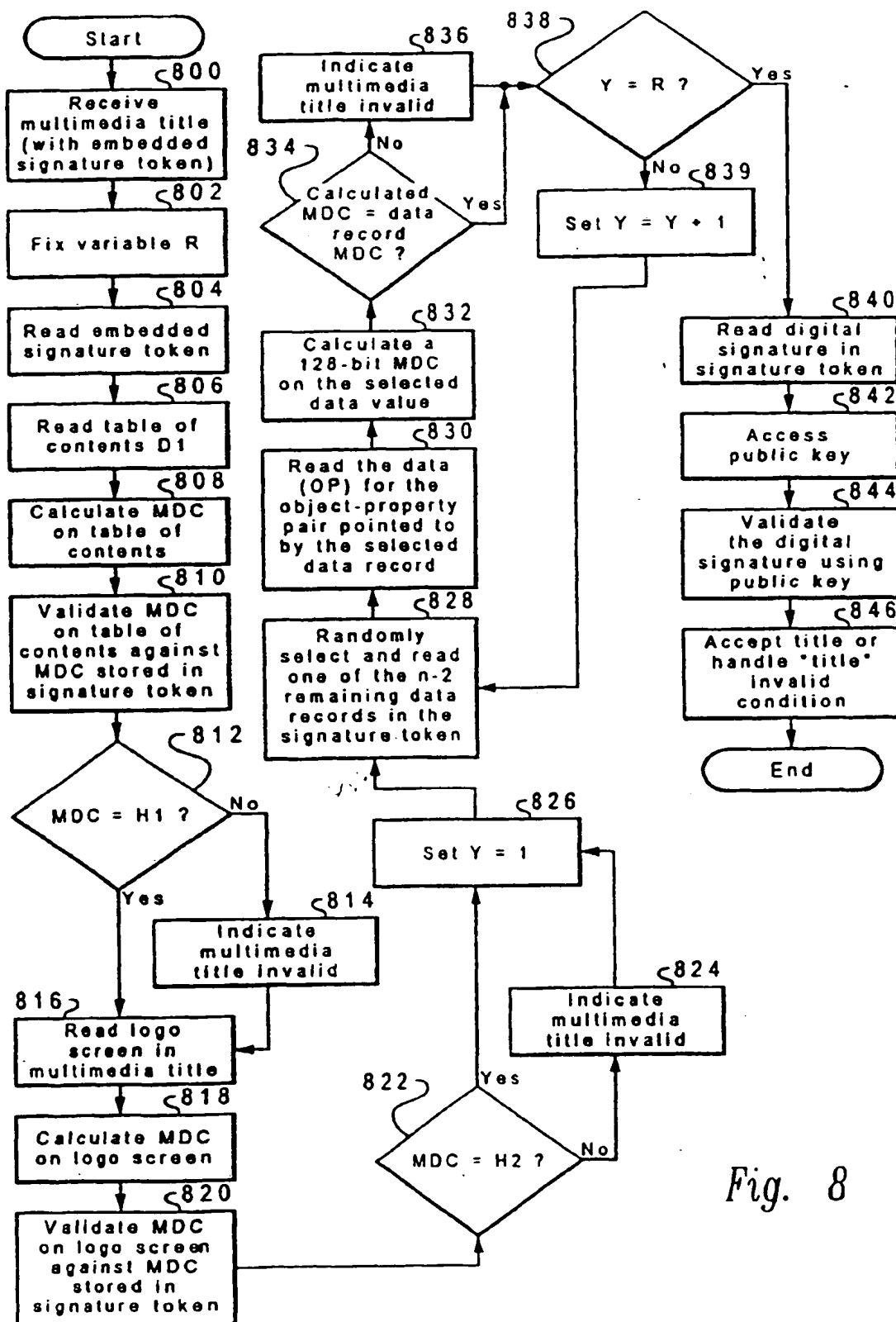
End

Fig. 7

Fig. 8

European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 95 30 8527

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	EP-A-0 565 314 (FISCHER ADDISON M) 13 October 1993 * figures 2,4,7,9,12,31 * * page 3, line 33 - line 42 * * page 9, line 2 - page 10, line 8 * * claims 1-47 *	1,4,5,9,10	G06F1/00 G06F12/14
A		6-8,11-13	
A	EP-A-0 570 123 (FISCHER ADDISON M) 18 November 1993 ---		
D,A	US-A-4 908 861 (BRACHTL BRUNO O ET AL) 13 March 1990		

TECHNICAL FIELDS SEARCHED (Int.Cl.6)

G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	29 March 1996	Powell, D

EPO FORM 1503 03.82 (P04C01)